O estudo utilizando apenas este material **não é suficiente** para o entendimento do conteúdo. Recomendamos a leitura das referências no final deste material e a resolução (por parte do aluno) de todos os exercícios indicados.

Grafos

Ordenação topológica

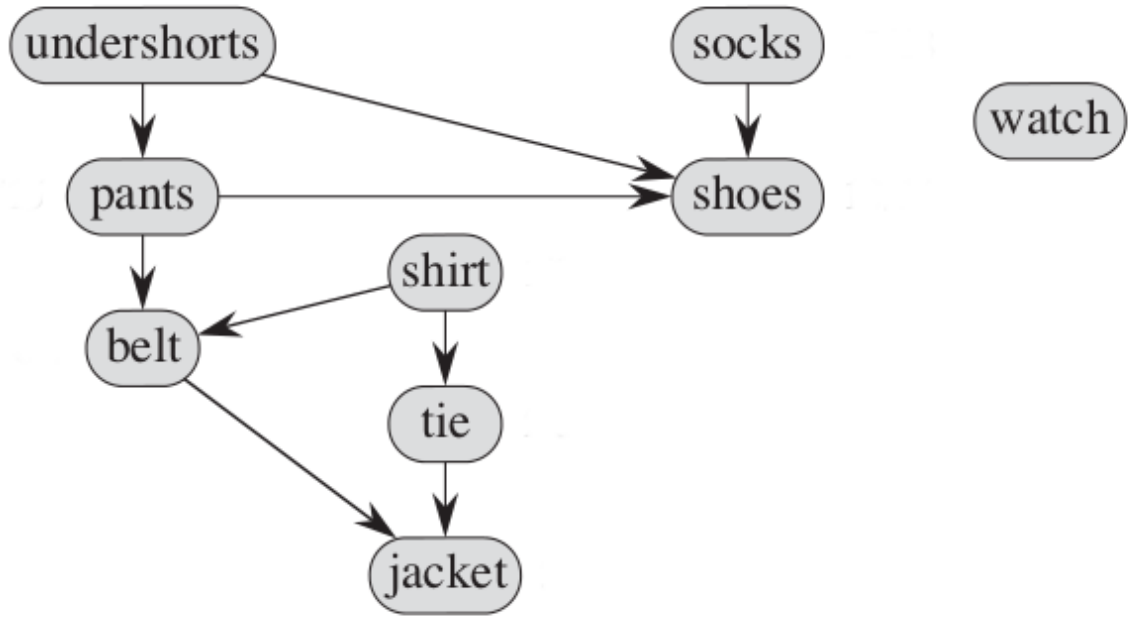
Conteúdo

Introdução

- Uma **ordenação topológica** de um grafo acíclico orientado $G = (V, E)$ é uma ordenação linear de todos os vértices, tal que para toda aresta $(u, v) \in E$, u aparece antes de v na ordenação
- Se os vértices forem dispostos em uma linha horizontal, todas as arestas devem ter a orientação da esquerda para direita
- Aplicação
  - Definição da ordem de execução de tarefas dependentes. Ex: `Makefile`

Ordenação topológica

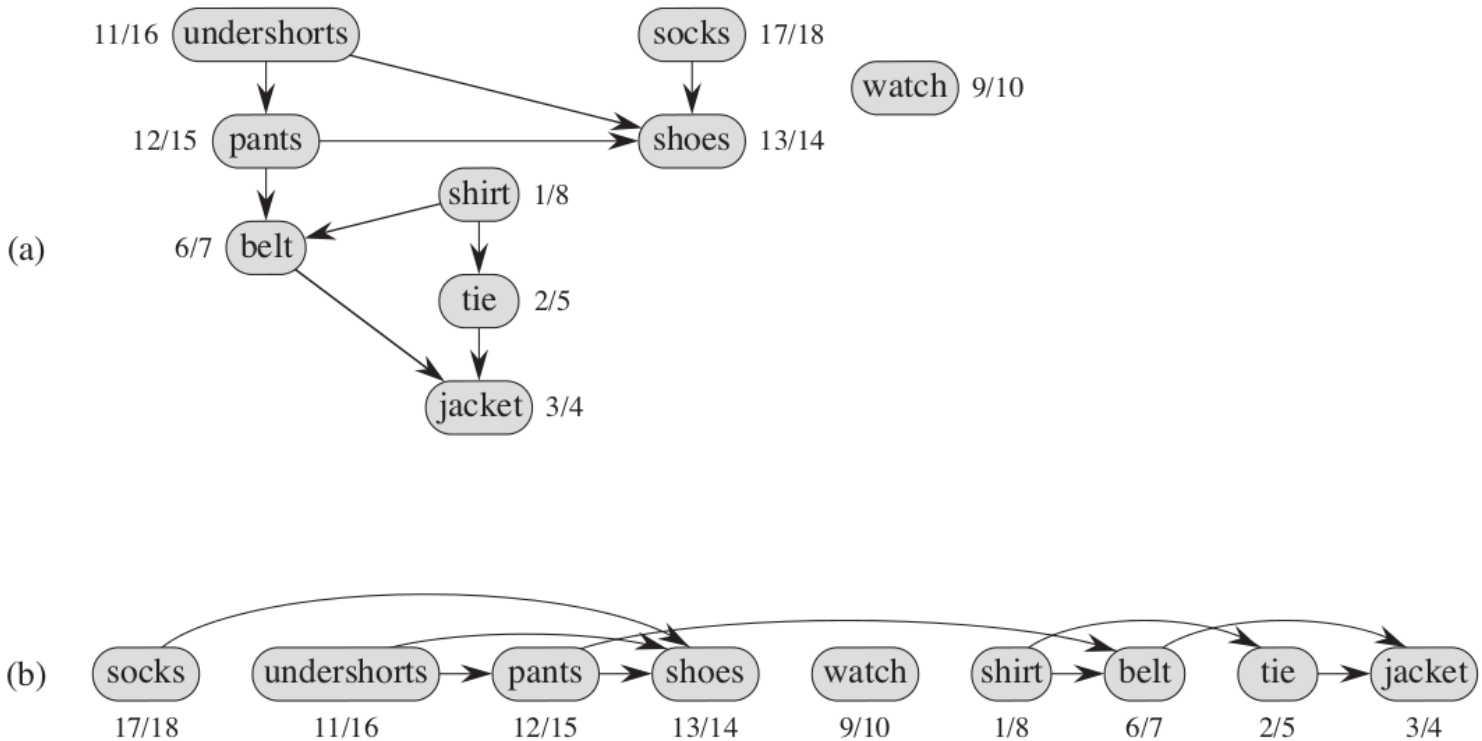
- Exemplo: o professor Bumstead deve se vestir pela manhã



Procedimento topological-sort

```
topological-sort(G)
1 chamar DFS(G) para calcular o tempo de término v.f
  para cada vértice v
2 à medida que cada vértice é terminado, inserir o
  vértice à frente de uma lista ligada
3 return a lista ligada de vértices
```

Exemplo de execução



Análise do tempo de execução `topological-sort`

- O tempo de execução da busca em profundidade é $\Theta(V + E)$
- O tempo para inserir cada vértice na lista de saída é $O(1)$, cada vértice é inserido apenas uma vez e portanto o tempo total gasto em operações de inserções é de $\Theta(V)$
- Portanto, o tempo de execução do algoritmo é $\Theta(V + E)$

Corretude do `topological-sort`

- Precisamos mostrar que se $(u, v) \in E$, então $v.f < u.f$
- Quando a aresta (u, v) é explorada, quais são as cores de u e v?
- u é cinza
- v é cinza também?
  - Não, porque isto implicaria que v é ancestral de u, e portando a aresta (u, v) seria uma aresta de retorno. Gaos não contém arestas de retorno
- v é branco?
  - Então v torna-se um descendente de u. Pelo teorema do parênteses $u.d < v.d < \mathbf{v.f} < \mathbf{u.f}$
- v é preto?
  - Então v já foi finalizado. Como a aresta (u, v) está sendo explorada, u não foi finalizado. Logo $v.f < u.f$

Exercícios

- 22.4-1 a 22.4-5

Exercícios

22.4-1 Mostre a ordenação de vértices produzidas por `topological-sort` quando ele é executado sobre o gao da figura 22.8. Considere os vértices em ordem alfabética, e suponha que cada lista de adjacência esteja em ordem alfabética.

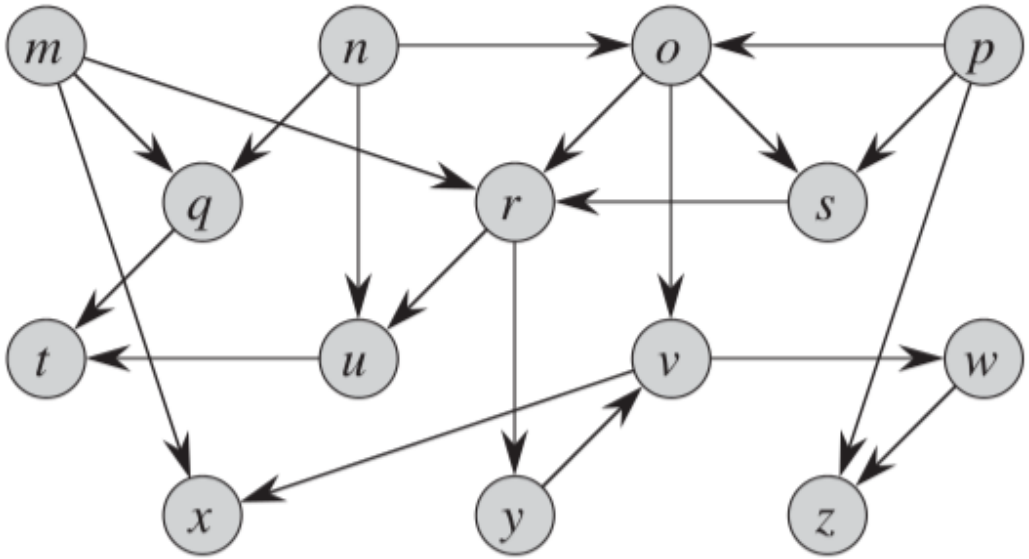


Figure 22.8 A dag for topological sorting.

Exercícios

22.4-2 Forneça um algoritmo de tempo linear que receba com entrada um grafo acíclico orientado $G = (V, E)$ e dois vértices s e t, e retorne o número de caminhos de s para t em G. Por exemplo, no gao da figura 22.8, existem exatamente quatro caminhos do vértice p para o vértice v: pov, $poryv$, $posryv$ e $psryv$. Seu algoritmo só precisa contar os caminhos, não listá-los.

22.4-3 Forneça uma algoritmo que determine se um dado grafo não orientado $G = (V, E)$ contém um ciclo. Seu algoritmo deve ser executado no tempo $O(V)$, independente de E.

Exercícios

22.4-4 Prove ou conteste: se um grafo orientado G contém ciclos, então `topological-sort(G)` produz uma ordenação de vértices que minimiza o número de arestas "ruins" que são incompatíveis com a ordenação produzida.

22.4-5 Outro modo de executar a ordenação topológica sobre um grafo acíclico orientado $G = (V, E)$ é encontrar repetidamente um vértice de grau de entrada 0, colocá-lo na saída e removê-lo do grafo, bem como todas as suas arestas de saída. Explique como implementar essa ideia, de tal forma que ela seja executada no tempo $O(V + E)$. O que acontecerá a esse algoritmo se G tiver ciclos?

Referências

- Thomas H. Cormen et al. Introduction to Algorithms. 3rd edition. Capítulo 22.4.